

chroot(2/8) and jail(2/8)

February 25, 2003

Adrian Filipi-Martin
adrian@ubergeeks.com

<http://www.chuug.org/talks/>

The Basics

Purpose:

- Restrict user/process view of a system.

Common Traits:

- Only available to the super-user.
- The apparent root of the filesystem is changed.
- A single process is affected and the attribute is then inherited by subprocesses.
- Once changed, the true filesystem root becomes permanently inaccessible. (Note, not really true for chroot. See below.)

chroot(8)

chroot path [command]

Starts the given "command", /bin/sh by default, with its view of the filesystem root changed to "path".

All shared libraries needed by a program must be available under this hierarchy with the correct paths.

Using statically linked programs can be easier.

Weakenesses of chroot(2)-based Jails

- root can get out using `fchdir(2)`.
- UID's and GID's are the same insided and out.
- chroot'ed processes can see all system processes.
- chroot'ed users can affect any processes they own or share a UID with.
- Users with access to a compiler or shell are dangerous.
- A process bound to a specific IP and port can steal traffic.

chroot(2) sysctl's

- `kern.chroot_allow_open_directories`

If set to zero, `chroot(2)` will always fail with `EPERM` if there are any directories open.

If set to one (the default), `chroot(2)` will fail with `EPERM` if there are any directories open and the process is already subject to a `chroot(2)` call.

Any other value will bypass the check for open directories.

This `sysctl` deals with the fact that `fchdir(2)` can be used to escape a jail if the `chroot'ed` process is holding open file descriptors to directories outside of the jail.

This limitation is not available on most platforms. Check your local `manpages`.

jail(8)

jail path hostname ip-address command ...

Same as for chroot(2) plus the following.

A jail is bound to an IP address. Typically an alias on the primary interface.

A jail has a hostname as well which should match the jail IP address.

There is no default command.

A jail exists as long as there are processes in it.

jail(2) Strengths

- jails are distinct from one another and the host system as well.
- Jailed users, including root, cannot affect processes outside of the jail that happen to have the same owner UID's.
- Process ownership is effectively a 2-tuple of (uid, jail-id).
- Jailed root users cannot change sensitive sysctl(2/8) values.

jail(2) Strengths (Cont.)

- Jailed processes cannot create raw sockets.
- Jailed processes cannot see external sockets, processes, etc.
- Jailed processes only send/receive traffic for the jail's IP.
i.e. A jailed process cannot steal traffic from the hosting system.
- Easily used to create virtual hosted machines.

jail(2) Weaknesses

- One IP per jail is required.
- You can have **ONLY** one IP per jail without 3rd party patches.
- Not portable to non-FreeBSD platforms.

jail(2) sysctl's

- **jail.set_hostname_allowed**

This MIB entry determines whether or not processes within a jail are allowed to change their hostname via `hostname(1)` or `sethostname(3)`

- **jail.socket_unixiproute_only**

The jail functionality binds an IPv4 address to each jail, and limits access to other network addresses in the IPv4 space that may be available in the host environment. This setting allows access to more than just the `PF_LOCAL`, `PF_INET`, and `PF_ROUTE` domains.

- **jail.sysvipc_allowed**

This MIB entry determines whether or not processes within a jail have access to System V IPC primitives.

Related Utilities

- **ldd(1)**

Use this to determine what shared libraries you need in your chroot'd/jailed environment.

- **restricted shells**

Sendmail's smrsh, ksh's rsh and bash's rbash. These prevent users from doing much, which limits the risk

- **sysctl(8)**

Use to make systemic environmentack changes as described above.

- **ftp daemons**

When you want to ensure that a message is only seen by the intended recipient.

Keep Non-jailed Users Out of a Jail

If setting up a host system with jails, be careful about sharing users, not UIDs but people, with the jailed environments. If a user has root in a jail and can access the jail from outside, it is trivial to hack the host system.

To prevent this, create a "firewall directory" that non-root users of the host system cannot cross. e.g.

```
install -m u=rwx,o= -d /usr/jails
install -m u=rwx,o= -d /usr/jails/jail1
install -m u=rwx,o= -d /usr/jails/jail2
etc...
```